Windows Message in the Unity3D : WndProc

先前提到可以用 Hooks 的方法在 Unity 裡監控 Windows Message，但是使用 Hooks 這個方法我們沒辦法更改 Message 的內容。因此這邊提出第二個方式。Windows 傳送 Message 給 Unity 時，會呼叫 Unity 預設的 Message 處理函數，但透過函數：  
  
pOldWndProc =(WNDPROC)SetWindowLong(hWnd, GWL\_WNDPROC, (LONG)SubWndProc);  
  
我們可以將原本 Windows 呼叫 Unity 預設的 Message 處理函數改為呼叫我們指定的函數，指定的函數把我們想要處理的 Message 處理完，再把剩下的 Message 丟回給 Unity 來處理。  
  
原本實作是使用 dllimport 讓 SetWindowLong function 可以在 C# 裡呼叫, 並把整個 callback function 及流程實作出來，且測試時運作都相當正常，但程式在關閉時會出現 Access Violation 的錯誤，後來將整個實作改成 C DLL 之後錯誤才沒有出現，不知道是什麼原因？底下是 DLL 部分的原始碼：  
  
#include "stdafx.h"

LRESULT CALLBACK SubWndProc(

HWND hWnd,

UINT nMessage,

WPARAM wParam, LPARAM lParam);

WNDPROC  gOldWndProc = NULL;

HWND  gUnityWnd = NULL;

#ifdef    \_\_cplusplus

extern "C" {

#endif    /\*    \_\_cplusplus    \*/

\_\_declspec(dllexport) bool \_\_stdcall init(HWND hWnd)

{

gOldWndProc =(WNDPROC)SetWindowLong(hWnd, GWL\_WNDPROC, (LONG)SubWndProc);

gUnityWnd =hWnd;

if (gOldWndProc !=NULL)

return true;

return false;

}

\_\_declspec(dllexport) void \_\_stdcall release()

{

SetWindowLong(gUnityWnd, GWL\_WNDPROC, (LONG)gOldWndProc);

gOldWndProc =0;

gUnityWnd =0;

}

#ifdef    \_\_cplusplus

}

#endif    /\*    \_\_cplusplus    \*/

LRESULT CALLBACK SubWndProc(

HWND hWnd,

UINT nMessage,

WPARAM wParam, LPARAM lParam)

{

switch(nMessage)

{

case WM\_IME\_SETCONTEXT:

case WM\_IME\_STARTCOMPOSITION:

case WM\_IME\_ENDCOMPOSITION:

case WM\_IME\_COMPOSITION:

case WM\_IME\_REQUEST:

{

//...

}

break;

}

return CallWindowProc(gOldWndProc, hWnd, nMessage, wParam, lParam);

}

Unity 可以透過呼叫 DLL 提供的 init() 函數，讓 Windows 改為呼叫我們指定的函數 (SubWndProc) 來處理 Message，透過 release() 函數讓 Message 處理流程復原。底下是 Unity 部分的原始碼（DLL 檔名為 UnityIMEDLL.dll 且檔案放在 Assets/Plugins 目錄下）

using UnityEngine;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Runtime.InteropServices;

public class IMEInputBox : MonoBehaviour

{

    //-----------------------------------------------------------

    [DllImport("UnityIMEDLL")]

    protected static extern bool init(IntPtr hWnd);

    [DllImport("UnityIMEDLL")]

    protected static extern void release();

    [DllImport("user32")]

    protected static extern IntPtr GetActiveWindow();

    //-----------------------------------------------------------

// Use this for initialization

void Start ()

{

        Debug.Log("init UnityIMEDLL.");

        try

        {

            init(GetActiveWindow());

        }

        catch (Exception e)

        {

            Debug.Log(e.ToString());

        }

}

    void OnDisable()

    {

        Debug.Log("release UnityIMEDLL.");

        try

        {

            release();

        }

        catch (Exception e)

        {

            Debug.Log(e.ToString());

        }

    }

}